**Summary**

My approach was to first make sure I had each element of the requirements in some basic form to ensure that I was covering each requirement. Afterwards I added the functionality of each test, whether it was restricting the number of characters the string could have or making sure that only the name and description were updateable within the task class and not the task ID. I have listed five example tests that help ensure good coverage throughout the project, I have not included them all since the list would be to long for this report.

@Test

**public** **void** testAddTaskPass() {}

@Test

**public** **void** testUpdateNamePass() {}

@Test

**public** **void** testUpdateNameFail() {}

@Test

**public** **void** testUpdateDescriptionPass() {}

@Test

**public** **void** testUpdateDescriptionFail() {}

My overall coverage for my JUnit tests for task ,task test, task services, task service test, contact ,contact test, contact services, contact services test, appointment test, appointment services and appointment services test were 100%, but the coverage for appointment was only 93.5%. These coverages show that I have a great overall quality of my JUnit test since they cover a total of 99.7% of the mobile application.

**Reflection**

The main testing technique that I employed was Unit testing. Unit testing is the first level of testing developers use. This process tests that each component in a piece of software function as it was designed to function. It also helps eliminate future errors by finding issues in the code early in the development process.

I did not use Integration testing or System testing techniques for this project since integration testing consists of testing whole segments of an application such as opening the application or logging in to the application. Doing integration testing would be used in testing if the GUI is properly displaying the correct information or if the system allows you to login successfully.

System testing is usually conducted by an outside team. It tests the entire system from end to end to ensure everything functions as expected before the system is put into production. The system test is a final once over test to ensure everything within a project or application is working correctly.

**Mindset**

There were several times throughout this project that I had to reassess how I was applying these tests to ensure that everything was working correctly and that I had sufficient coverage throughout the entire project. I need to slow down and realize that there were many aspects of the code that I was not covering to its fullest extent and need to make sure that I included testing for those specific functions, such as the toString() function within the contact class.

I tried to make sure that I was unbiased when it came to assessing my code for when it came to the testing of that code. As mentioned in the example toString() function above. In my initial test of the contact class functionality, I excluded that test for toString() since I felt at the time that it was not crucial to the function of the program. I realized that if that function was not tested properly then an unwanted output could show the client that the quality of the program was not up to par, and they could decide to go a different route with another competing company or scrap the entire idea with us which would end up cost our company a lot of money if that happened. So, it is important to your career and to the company that you are employed by that you do not cut corners when it comes to testing and writing code.

To avoid future possible technical debt I plan on reviewing the entirety of my code and tests for that code to ensure that I achieve 100% coverage throughout the program so that I can be not only sure that the program will work as it is intended to work but that I can be proud of the quality of work that I accomplished.